University of Tripoli
Faculty of Information Technology

Department of Software Engineering

CSS3 Grid

Introduction to Internet Programming
ITGS 226 -- F 2021

By: Fatima Ben Lashihar

BASICS

- CSS Grid Layout is a two-dimensional grid-based layout system that, compared to any web layout system of the past, completely changes the way we design user interfaces.
- Grid is one of the most powerful CSS modules ever introduced.

2

GRID TERMINOLOGIES

- **Grid Container:** The element on which display: grid is applied. It is the direct parent of all the grid items.
- **Grid Line:** The dividing lines that make up the structure of the grid. They can be either vertical (“column grid lines”) or horizontal (“row grid lines”) and reside on either side of a row or column.
- **Grid Track:** The space between two adjacent grid lines. It can be thought as the columns or rows of the grid.
- **Grid Area:** The total space surrounded by four grid lines. It may be composed of any number of grid cells.

GRID TERMINOLOGIES cont’d

- **Grid Item:** The children (i.e. *direct* descendants) of the grid container.
- **Grid Cell:** The space between two adjacent row and two adjacent column grid lines. It’s a single “unit” of the grid.
- **fr units:** fractional units which essentially mean “portion of the remaining space”.
- **auto**: this keyword is a lot like fr units, except that they “lose” the fight in sizing against fr units when allocating the remaining space.
- **The repeat() function:** can save some typing.

GRID PROPERTIES

- Properties for the Grid container
- Properties for Grid items

PROPERTIES FOR THE PARENT (GRID CONTAINER)

- **Display:** defines the element as a grid container and establishes a new grid formatting context for its contents.

```
.container {
      display: grid | inline-grid;
```

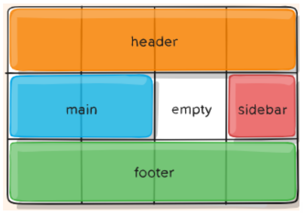
- **grid-template-columns**
- **grid-template-rows**

Defines the columns and rows of the grid with a space-separated list of values. The values represent the track size, and the space between them represents the grid line.

PROPERTIES FOR THE PARENT (GRID CONTAINER) Cont'd

- **grid-template-areas:**

Defines a grid template by referencing the names of the grid areas which are specified with the (grid-area property). Repeating the name of a grid area causes the content to span those cells. A period signifies an empty cell.



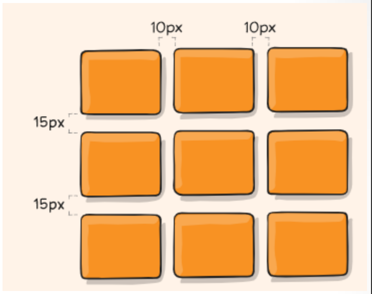
```
.item-a { grid-area: area1; }
.item-b { grid-area: area2; }
.item-c { grid-area: area3; }
.item-d { grid-area: area4; }
.container {
      display: grid;
      grid-template-columns: 50px 50px 50px 50px;
      grid-template-rows: auto;
      grid-template-areas:
              "area1 area1 area1 area1 "
              "area2 area2   .   area3"
              "area4 area4 area4 area4 ";
 }
```

7

PROPERTIES FOR THE PARENT (GRID CONTAINER) Cont'd

- **column-gap**
- **row-gap**

Specifies the size of the grid lines. These gaps are only created *between* the columns/rows, not on the outer edges.



```
.container {
      grid-template-columns: 100px 50px 100px;
      grid-template-rows: 80px auto 80px;
      column-gap: 10px;
      row-gap: 15px;
}
```

8

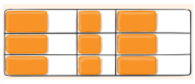
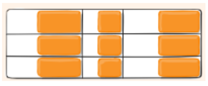
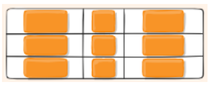
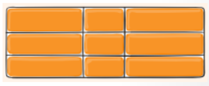
PROPERTIES FOR THE PARENT (GRID CONTAINER) Cont'd

- **justify-items:** Aligns grid items along the *inline (row)* axis . This value applies to all grid items inside the container.

Values:

```
.container {
      justify-items: start | end | center | stretch;
 }
```

Where stretch is the default.

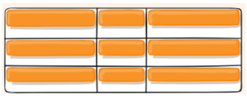
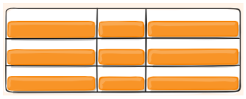
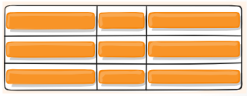
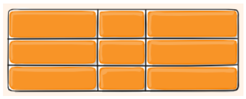





PROPERTIES FOR THE PARENT (GRID CONTAINER) Cont'd

- **align-items:** Aligns grid items along the *block (column)* axis. This value applies to all grid items inside the container.

Values:

```
.container {
      align-items:start |end |center |Stretch;
  }
```

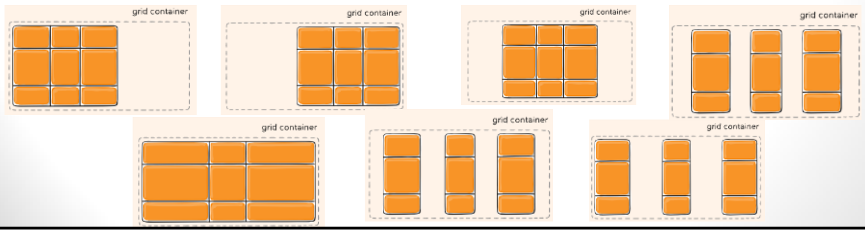





PROPERTIES FOR THE PARENT (GRID CONTAINER) Cont'd

- **justify-content:** aligns the grid along the *inline (row)* axis. It happens when the total size of the grid might be less than the size of its grid container (grid items are sized with non-flexible units like px).

Values:

```
.container {
  justify-content: start | end | center |
                   stretch | space-around |
                   space-between | space-evenly; }
```

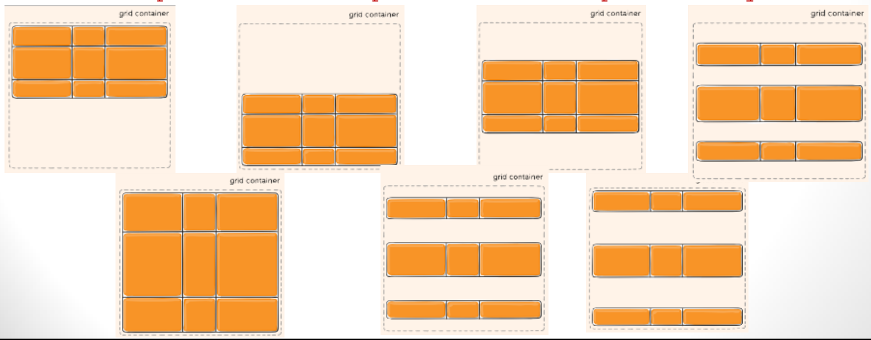


PROPERTIES FOR THE PARENT (GRID CONTAINER) Cont'd

- **align-content:** aligns the grid along the *block (column)* axis (as opposed to justify-content

Values:

```
.container {
   align-content: start | end | center | stretch |
      space-around | space-between | space-evenly; }
```



PROPERTIES FOR THE PARENT (GRID CONTAINER) Cont'd

- **grid-auto-columns**
  grid-auto-rows
- Specifies the size of any auto-generated grid tracks

- **grid-auto-flow:** controls how the auto-placement algorithm works.
- Values:

.container { grid-auto-flow: row | column; }

PROPERTIES FOR THE CHILDREN (GRID ITEMS)

- **grid-column**
- **grid-row**

Determines a grid item's location within the grid by referring to specific grid lines.

Values:

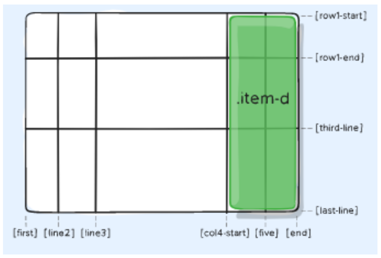
```
.item {
      grid-column: <start-line> / <end-line>;
      grid-row: <start-line> / <end-line>;
}
```

If no end line value is declared, the item will span 1 track by default.

PROPERTIES FOR THE CHILDREN (GRID ITEMS) Cont'd

- **grid-area:** Gives an item a name so that it can be referenced by a template created with the grid-template-areas property.

```
.item-d { grid-area: header; }
.item-d {
     grid-area: 1 / col4-start / last-line / 6;
}
```



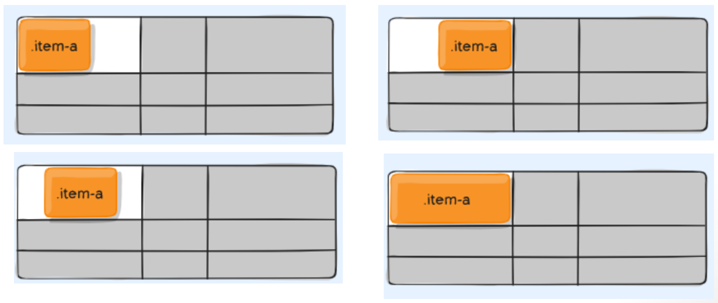
15

PROPERTIES FOR THE CHILDREN (GRID ITEMS) Cont'd

- **justify-self:** Aligns a grid item inside a cell along the *inline (row)* axis. This value applies to a grid item inside a single cell.

Values:

.item { justify-self: start | end | center | stretch; }



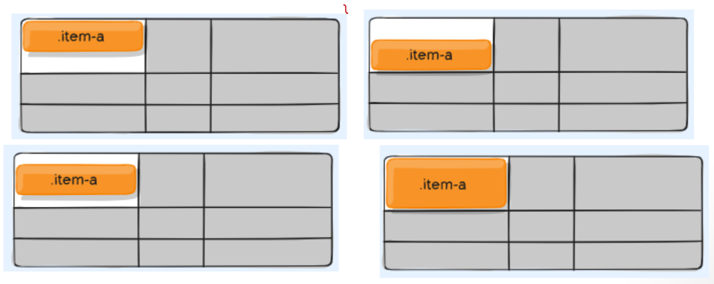
16

PROPERTIES FOR THE CHILDREN (GRID ITEMS) Cont'd

- **align-self:** Aligns a grid item inside a cell along the *block (column)* axis (as opposed to justify-self which aligns along the *inline (row)* axis).

Values:

```
.item {
      align-self: start | end | center | stretch;
}
```



17

THE END

18